VZ-LINK MAGAZINE

NEW ZEALAND

VZ 300/200

COMPUTER USERS CLUB

January

19 90

ISSUE 39

HAPPY NEW YEAR

FROM ME TO YOU

Well, we have made it to another issue.I am hoping that 1990 will see the VZ club continue.

I have been flat out over the last couple of week's,what with helping my wife pack/move and also starting work again.

I have had to hold over the Questionare until next issue.
This issue is packed with a little something for everyone.

WARNING.

DO NOT remove any RAM packs or interfaces from your computer until you switch off the power.You can cause a lot of damage if you do.

POOR SERVICE.

Some two months back I returned my Speech Syn circuit boards to MINITECH because they were not done correctly.
To date I have not received these boards back.Endless phone calls have resulted in excuse after excuse.

I suggest that if these companies do not wish to help the hobbiest they should come right out and say so.Instead of taking our money and placing us on the bottom of their customer list.

The only thing I can suggest is you avoid these companies. Try and do it yourself or ask other hobbiest for help.

REMEMBER.

If you want programs or help with anything,feel free to ring me on 761-451.

If I can't help you,I can put you on to someone that can.

Here are a few suggestions for a more interesting newsletter:

1) A "Letters to the editors" Section.

2) A "How do I get this program to work" Section.

3) A "How do I convert this program for VZ use" section.

4) A "How do I interface the VZ" section.

Have a think about these suggestions and let me know what you think or if you would be interested in helping.

Any other suggestions welcome.

Well that's all for this month,but I should be back to a on time newsletter next month (I hope).

NOTES AND ERRATA

IN ISSUE 38 OF DECEMBER 89
THE MORSE LEARNING PROGRAM WRITTEN BY STEVE DALE HAS A ERROR IN ONE OF THE LINE.
LINE 1380 SHOULD READ :-

```
1380 PRINT@390,G$;" IS WRONG":SOUND10,3:H=H+1:GOSUB1390:GOTO1265
```

THANK YOU STEVE FOR POINTING THIS OUT

```
370 CLS:FORJ=1TOL:FORI=1TOLEN(Z$(J))
```

This line contains two loops. In computer jargon when one loop is **INSIDE** another it is called a NESTED loop.
The **INNER** loop (I) will run all its steps for every step of the **OUTER** loop (J).
In this case, while the **OUTER** loop is selecting each line of text in turn, the INNER loop will be selecting every character in each line of text in turn.
In this way we can examine every character that has been typed and change its ASCII number if necessary, before passing it on to the printer.
Seem a big task, but don't worry, your computer does it very quickly.
L is the number of text lines.
Z$(J) is each line in turn, while **LEN (Z$(J))** is the length (or the number of character) in each line.

```
380 A=ASC (RIGHT$(LEFT$(Z$(J),I),1))
```

IMPORTANT It is important that you type this line correctly. Please check that all the BRACKETS and COMMAS are exactly as shown here.

If Z$(1) is the first line of text, then **LEFT$(Z$(J),1)** will be the first character of that line. **LEFT$(Z$(J),2)** will be the first TWO characters. Now, if we apply a **RIGHT$(...)** to those two letters, then the **RIGHT$** will be only the second character. So, if we make **A** the **ASC**, (ASCII number) of the **RIGHT$** of the **LEFT$** then **A** will become the ASCII number of each character in turn.

```
390 IFA>=64ANDA<=95THENA=A+32
400 IFA>=192ANDA<=223THENA=A-128
410 IFA>=224ANDA<=255THENA=A-192
```

Line 390 says, if **A** (which will be the ASCII number of each character of the text in turn) is **EQUAL TO** or **GREATER THAN 64**, AND if **A** is **EQUAL TO** or **SMALLER THAN 95**, then ADD 32 to **A**. The block of ASCII numbers for 32 to 95 are held in computer Memory as ordinary letters, numbers and punctation marks. By "ordinary" we mean **NOT INVERSE**. If they were **LPRINTed** the Letters would be **CAPITAL** letters. We had decided that **VeeZed INVERSE** letters would be **UPPER** case. These ordinary letters are to be used for **LOWER** case. By increasing the value of **A** by 32 in this block then they will be the ASCII numbers accepted by the printer as **LOWER** case letters. Simple, eh?

Line 400 deals with the block of the **VeeZeds INVERSE** characters, changing them by subtracting 128 so they will **LPRINT UPPER** Case. And line 410 does a similar change to **INVERSE** numbers and punctuations so that these can be entered in **INVERSE** or as ordinary characters. The main point of this line is that this block includes a **COMMA**, a **COLON** and a **SEMI-COLON**.
These three characters, when entered in text lines, causes the VeeZed to produce the error message **EXTRA IGNORED**, so the only way round the problem was to only use them in **INVERSE**. This line does that.

```
420 REM
430 REM
```

These lines are for sending Printer Control Codes to change the status of a printer. Some printers can be made to print ITALICS, in SUPERSCRIPT and SUBSCRIPT, or even to UNDERLINE. Obviously, we cannot put in **Printer Controls** unless we know what printer is to be used.

```
440 LPRINTCHR$(A);:NEXT
```

LPRINTCHR$(A); because it is followed by a **SEMI-COLON** does **NOT LPRINT**. Instead, it loads the value of **A** into the printer's memory, which is called the **BUFFER**. **NEXT**, of course, sends the **INNER** loop (I) back for another **A**.

```
450 LPRINT:PRINTZ$(J):NEXT:GOTO230
```

This LPRINT does LPRINT. It releases all the characters stored in the BUFFER of the printer so that a whole line is LPRINTed. NEXT continues the OUTER loop (J) which goes back for another line of text.
PRINTZ$(J) comes before NEXT so that the line, as typed, will be printed on the screen, keeping you informed of the progress of this sub-routine.
Finally, GOTO230 sends you back to the MENU.

EDITING SUB-ROUTINE

This sub-routine may appear complex. Don't worry if my explanation leaves you starry eyed. Instead of trying to explain it in detail, I will just give a general indication of what it is doing.

```
460 CLS:PRINT@40,"EDIT":SOUND14,3:
PRINT@45,"LINE";:INPUTC:X=C
470 IFC>LTHEN460
```

NOTE Underlined characters are to be entered in INVERSE.

Line 460 after clearing the screen, prints the word EDIT after a short SOUND, and adds the word LINE. The INPUT puts a question mark after LINE, making it an invitation to enter the line number you wish to Edit. X is made equal to C. X will be used in line 270 of the Menu to show which line was last edited. Line 470 says if the line number entered is MORE THAN the the number of lines in the text, (L), then go back to 460 and try again.

```
480 PRINT@0,"LINE"C
490 PRINT@32,"      ZZZZZZZZZZZZZZZZZZZZZZZZZZ"
500 PRINT@192,Z$(C)
510 PRINT@128,"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
520 PRINT@32,"DELETE":INPUTD$
530 B=64:FORD=0TO63:PRINT@B," ":B=B+1:NEXT
540 IFD$=""THEN230
```

Line 480 prints the line number at the top of the screen.
Line 490 prints lines similar to the Typing Guides. There are six spaces, (where the word DELETE is going), followed by 26 SHIFTZ or SHIFTJ, if you prefer, Graphics. Then line 500 will print the line requested for editing. Line 510 prints 32 SHIFT Zs (or SHIFTJs) and 520 adds the word DELETE and puts an INPUT question mark between the Guide Lines. This is where you type the letter or letters, word or words that have to be DELETED.
Line 530 wipes out whatever you've entered in preparation for you to type the correction. Line 540 is there to return you to the menu if you have entered nothing at all.

```
550 Q=LEN(D$):S=LEN(Z$(C))
560 SOUND30,1:PRINT@32," ENTER":INPUTR$
```

Q becomes the length of your DELETE, and S becomes the length of the original line of text. Q and S will be used shortly.
Line 560 prints ENTER over the top of DELETE. Because ENTER is one letter short we have added an INVERSE SPACE in before the letter E. The INPUT puts a question mark inside the Guides.
This time you type in the letter or letters, word or words to be ENTERed in place of those to be DELETEd.
It is important that the letter(s) or word(s) to be deleted are EXACTLY as originally typed or the computer will not be able to recognise them.
Using the LEN(...) information the computer is going to look for your DELETE characters. If it cannot find them it will let you know.

```
570 FORZ=1TOS
580 IFMID$(Z$(C),Z,Q)=D$THEN620
590 NEXT
600 PRINT@97,"INCORRECT"
610 SOUND4,9:GOTO230
```

The loop is for finding the character(s) to be deleted. If the computer finds them it will send the program to line 620, but if it does not find them it prints the word INCORRECT and after blowing you a *raspberry* puts you back to the menu.

```
620 E=Z-1+LEN(D$)
630 N$=LEFT$(Z$(C),Z-1)+R$+RIGHT$(Z$(C),LEN(Z$(C))-E)
640 Z$(C)=N$
```

This is another place where you should be careful to get all those COMMAS and BRACKETS correct. This is where the computer uses a temporary String Variable (N$) to build the new line of text by replacing your DELETE with your ENTER.

```
650 FORD=192TO256:PRINT@D,CHR$(32):NEXT
660 PRINT@192,Z$(C):FORT=1TO1000:NEXT:GOTO230
```

Line 650 erases the original line of text from the screen. This method is just an alternative to the method used in line 530. CHR$(32) is the same as " " (just a SPACE).
Line 660 prints the new EDITED line of text with a FOR/TO loop to provide sufficient time for you to see if it is correct before going back to the menu.

All that remains now is the sub-routines for FILING TO TAPE, RETRIEVING FROM TAPE and SCROLLING the text up the screen.

IT MIGHT BE A GOOD IDEA NOW TO TEST RUN, DEBUG AND CSAVE.

TAPE FILE

```
670 CLS:PRINT@68,"TAPE FILE":SOUND16,2:PRINT@235,"RECORD"
680 SOUND22,9:CLS
690 FORJ=1TOL
700 PRINT#"",Z$(J):PRINTZ$(J):SOUND30,1:NEXT
710 CLS:PRINT@235,"STOP TAPE":SOUND24,9:CLS:GOTO230
```

Line 670. After clearing the screen the words TAPE FILE are printed and then followed by RECORD. A couple of SHIFTJs either side of RECORD can help to dress up the screen display.
Line 680. The long sound provides the opportunity for the tape to be started on RECORD.
Line 690. Here we begin the loop for recording the text from line 1 to line L, which is the total number of lines entered.
Line 700. This is the command for recording to tape.
The quotes normally contain a FILENAME, but as we cannot use a String Variable to change the Filename each time, we have left the quotes empty. As we have done the same in the RETRIEVE sub routine, it means that this program cannot be used to recover data that has been recorded with a Filename, unless the Filename is added.
As each line of text is recorded it is printed on the screen with a short SOUND.
Line 710. When the recording is completed STOP TAPE appears. A long sound acts as a warning that the recording is finished. Then back to the menu.

RETRIEVE FROM TAPE.

```
720 CLS:PRINT@226,"RETRIEVE":SOUND22,6:PRINT@290,LINES";:
    INPUTL
730 CLS:SOUND24,5
740 FORJ=1TOL:INPUT#"",Z$(J):PRINTZ$(J):NEXT
750 PRINT@490,"STOP TAPE":SOUND26,9
760 GOSUB790:GOTO230
```

Line 720. Clear the screen, print the title RETRIEVE and after a SOUND print the word LINES. The INPUT puts a question mark after the word LINES inviting you to enter the number of lines to be retrieved.
Line 730 clears the screen and the SOUND tells you it is time to start the tape on PLAY.
Line 740. Here again is the loop which runs from line one to L which is the total number of lines of text to be retrieved. Each line appears on the screen as it is retrieved.
Line 750. STOP TAPE is printed on the screen with a long SOUND to let you know that the Retrieval is complete.
Line 760. GOSUB790 picks up a DELAY loop which is part of the SCROLL sub-routine then returns you to the menu.

SCROLL

```
770 CLS:FORJ=1TOL:PRINTJ:PRINTZ$(J):GOSUB790:NEXT:GOSUB790
780 GOTO230
790 FORT=1TO1600:NEXT:SOUND20,1:RETURN
```

SCROLL is to have your text printed on the screen line-by-line to give you the opportunity of checking the text for errors. The line number is printed just before the line itself so that you can make a note of which lines need to be corrected. The SCROLLING speed needs to be suited to your own reading speed. This is done by changing line 790. To make the SCROLL run more slowly just change 1600 to a larger number.
Line 770 is the now familiar loop running from text line 1 to L (the total number of lines). Firstly it prints on the screen the line number (J), then the line followed by GOSUB790 which is the DELAY loop. Another GOSUB790 follows NEXT to give you time to read the last line of text before getting back to the menu (line 780).

That is the end of the MICRO TYPE program except for making it suitable for whatever Printer or Plotter you will be using.

MICRO MAGIC is presented here in its most simple form. There is no provision for Printer Control codes.These codes were discussed in the Studio Ad Lib pages of the November issue of VZ LINK.

In the same issue we also gave details of how Printer Control Codes are sent from the VeeZed to printers.You should be able to adapt the MICRO TYPE program to suit your particular printer.

(If you have any difficulty just write,telling me what printer you have and what form your Printer Instruction book says the control codes take.
A photocopy would be even better.....peter)

CONVERSION FOR DISK SAVE AND RETRIEVE by Peter Hill
Replace and Add new lines as shown,All other lines remain the same.

DISK FILE

```
670 CLS:PRINT@68,"DISK FILE":SOUND16,2:PRINT@235,"INSE RT DISK"
680 SOUND22,9:CLS
682 OPEN"MT",1
690 FOR J=1TO L
700 PR#"MT",Z$(J):PRINTZ$(J) :SOUND 30,1:NEXT
710 CLS:PRINT@235,"FILE SAVE COMPLETE":CLOSE"MT":SOUND24,9: CLS:
GOTO 230
```

```
570 FORZ=1TOS
580 IFMID$(Z$(C),Z,Q)=D$THEN620
590 NEXT
600 PRINT@97,"INCORRECT"
610 SOUND4,9:GOTO230
```

The loop is for finding the character(s) to be deleted. If the computer finds them it will send the program to line 620, but if it does not find them it prints the word INCORRECT and after blowing you a *raspberry* puts you back to the menu.

```
620 E=Z-1+LEN(D$)
630 N$=LEFT$(Z$(C),Z-1)+R$+RIGHT$(Z$(C),LEN(Z$(C))-E)
640 Z$(C)=N$
```

This is another place where you should be careful to get all those COMMAS and BRACKETS correct. This is where the computer uses a temporary String Variable (N$) to build the new line of text by replacing your DELETE with your ENTER.

```
650 FORD=192TO256:PRINT@D,CHR$(32):NEXT
660 PRINT@192,Z$(C):FORT=1TO1000:NEXT:GOTO230
```

Line 650 erases the original line of text from the screen. This method is just an alternative to the method used in line 530. CHR$(32) is the same as " " (just a SPACE).
Line 660 prints the new EDITED line of text with a FOR/TO loop to provide sufficient time for you to see if it is correct before going back to the menu.

All that remains now is the sub-routines for FILING TO TAPE, RETRIEVING FROM TAPE and SCROLLING the text up the screen.

IT MIGHT BE A GOOD IDEA NOW TO TEST RUN, DEBUG AND CSAVE.

TAPE FILE

```
670 CLS:PRINT@68,"TAPE FILE":SOUND16,2:PRINT@235,"RECORD"
680 SOUND22,9:CLS
690 FORJ=1TOL
700 PRINT#"",Z$(J):PRINTZ$(J):SOUND30,1:NEXT
710 CLS:PRINT@235,"STOP TAPE":SOUND24,9:CLS:GOTO230
```

Line 670. After clearing the screen the words TAPE FILE are printed and then followed by RECORD. A couple of SHIFTJs either side of RECORD can help to dress up the screen display.
Line 680. The long sound provides the opportunity for the tape to be started on RECORD.
Line 690. Here we begin the loop for recording the text from line 1 to line L, which is the total number of lines entered.
Line 700. This is the command for recording to tape.
The quotes normally contain a FILENAME, but as we cannot use a String Variable to change the Filename each time, we have left the quotes empty. As we have done the same in the RETRIEVE sub routine, it means that this program cannot be used to recover data that has been recorded with a Filename, unless the Filename is added.
As each line of text is recorded it is printed on the screen with a short SOUND.
Line 710. When the recording is completed STOP TAPE appears. A long sound acts as a warning that the recording is finished. Then back to the menu.

RETRIEVE FROM TAPE.

```
720 CLS:PRINT@226,"RETRIEVE":SOUND22,6:PRINT@290,LINES";:
    INPUTL
730 CLS:SOUND24,5
740 FORJ=1TOL:INPUT#"",Z$(J):PRINTZ$(J):NEXT
750 PRINT@490,"STOP TAPE":SOUND26,9
760 GOSUB790:GOTO230
```

Line 720. Clear the screen, print the title RETRIEVE and after a SOUND print the word LINES. The INPUT puts a question mark after the word LINES inviting you to enter the number of lines to be retrieved.
Line 730 clears the screen and the SOUND tells you it is time to start the tape on PLAY.
Line 740. Here again is the loop which runs from line one to L which is the total number of lines of text to be retrieved. Each line appears on the screen as it is retrieved.
Line 750. STOP TAPE is printed on the screen with a long SOUND to let you know that the Retrieval is complete.
Line 760. GOSUB790 picks up a DELAY loop which is part of the SCROLL sub-routine then returns you to the menu.

SCROLL

```
770 CLS:FORJ=1TOL:PRINTJ:PRINTZ$(J):GOSUB790:NEXT:GOSUB790
780 GOTO230
790 FORT=1TO1600:NEXT:SOUND20,1:RETURN
```

SCROLL is to have your text printed on the screen line-by-line to give you the opportunity of checking the text for errors. The line number is printed just before the line itself so that you can make a note of which lines need to be corrected. The SCROLLING speed needs to be suited to your own reading speed. This is done by changing line 790. To make the SCROLL run more slowly just change 1600 to a larger number.
Line 770 is the now familiar loop running from text line 1 to L (the total number of lines). Firstly it prints on the screen the line number (J), then the line followed by GOSUB790 which is the DELAY loop. Another GOSUB790 follows NEXT to give you time to read the last line of text before getting back to the menu (line 780).

That is the end of the MICRO TYPE program except for making it suitable for whatever Printer or Plotter you will be using.

MICRO MAGIC is presented here in its most simple form. There is no provision for Printer Control codes.These codes were discussed in the Studio Ad Lib pages of the November issue of VZ LINK.

In the same issue we also gave details of how Printer Control Codes are sent from the VeeZed to printers.You should be able to adapt the MICRO TYPE program to suit your particular printer.

(If you have any difficulty just write,telling me what printer you have and what form your Printer Instruction book says the control codes take.
A photocopy would be even better.....peter)

CONVERSION FOR DISK SAVE AND RETRIEVE by Peter Hill
Replace and Add new lines as shown,All other lines remain the same.

DISK FILE

```
670 CLS:PRINT@68,"DISK FILE":SOUND16,2:PRINT@235,"INSE RT DISK"
680 SOUND22,9:CLS
682 OPEN"MT",1
690 FOR J=1TO L
700 PR#"MT",Z$(J):PRINTZ$(J) :SOUND 30,1:NEXT
710 CLS:PRINT@235,"FILE SAVE COMPLETE":CLOSE"MT":SOUND24,9: CLS:
GOTO 230
```

RETRIEVE FROM TAPE

```
735 OPEN"MT",0
740 FORJ=1TO L:IN#"MT",Z$(J):PRINTZ$(J):NEXT
750 CLS:CLOSE"MT":PRINT@490, "RETRIEVE COMPLETE":SOUND26,9
```

What is needed for this program is a SUB-ROUTINE That will allow different file names instead of just MT.
To do this you would need to be able to use a String Variable to change the Filenames.

This I will leave upto someone else to do.
Maybe You could write a little item for the newsletter telling us all how to do it.

DISK DRIVE ENQUIRY , AN ANSWER by L Whetton.

In answer to Lou Ellmers question in the November issue of VZ Link regarding the compatibility of APPLE disk drives with the VZ 300 System.

I have had my Two drives in a test system where the drives were plugged to the APPLE test socket on the drive test panel.

As far as the main Printed Circuit board is concerned the VZ 300 uses the same circuit as the APPLE with the same socket pins.

However the big difference is the speed of the disk drive motors.The VZ disk drive has been designed to run at a speed of 85 rpm which is much slower than the 300 rpm of most disk drive systems. Also ,iqnore the strobe pattern which is for 300 rpm.

As far as the VZ 300 drive is concerned the Printed circuit board for the disk drive motor is a special.I still have not been able to find a local source of supply for the speed regulator chip that is used for the speed control .
Also,not all of the APPLE disk drive control system is incorporated in the VZ disk drive,this being the reason for the head movement noise at switch on and no disk revolution count detection connected.

Hope this information helps...
L.WHETTON

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME IF YOU HAVE PROBLEMS
PHONE 761-451

SOFTWARE REVIEW

SOFTWARE REVIEW WHIZKID SPYCATCHER
by Brenda Hill

Whizkid spycatcher is one of a small batch of educational programs which were sold by D.S.E for the VZ.
And was written by Wayne Richmond in 1983.

I found that this program put's the fun back into learning maths.It is a good program for children from the age of eight,and dare I even say,for some adults who need to brush up on their basic maths.

There are 6 whiz kid soldiers (some are spies) waiting to cross the river.
The idea of the game is to stop the spies and allow the soldiers to get across.

At the beginning of the game you are asked to enter your name.You are then given instructions.

The soldiers and spies give you a sum along with the answer.You have to decide if the answer is correct or not, Using mental calculations.

The spies give you incorrect answers (they lie) and the soldiers give you the right answer (they tell the truth).
If you decide the answer to the sum is wrong you must trap the spy by dropping the trap on him using the space key.
On the other hand if you decide the answer to the sum is correct you let the soldier cross.

If you quess wrong and allow a spy to slip through,he laughs at you and you have to start the game again.
if you trap a soldier you also have to start the game again.
And if you manage to catch the spy,you are given the correct answer to the sum and then continue.

There are eight levels to this game,starting from level eight which is the easiest.
The lower the level the harder it get's.As you finish each level you are asked to hit "RETURN"if you wish to move on to the next level.

CONCLUSION
This game is well written,as far as it goes.
I find it does help me brush up on my maths using mental calculations, instead of working it out on paper.
I can also see this program being very useful as my children get a little older.

There are one or two little things about the program which could be improved.
Such as the program asking for your name and giving instructions each time you have to start again.This could be contained in a sub-routine and used once.
If you have more than one player this can be worked with another sub-routine.
Otherwise,as stated,this is a fun way of learning with a program that has great potential.....6 out of 10.

CSAVE/CLOAD BY DAVE MITCHELL

There are a few routines around that save and load screen dislpays from tape but they involve moving the displays around in memory using basic. They do but are a bit messy where you CLOAD the display but after it is loaded the CLOAD routine jumps to basic and does not return to the program you are running. You then have to run the next line number after CLOAD.

What should be done is to run the program without any breaks. This is what I have achieved. The program CLOADS MODE (1) displays from tape and loads directly to the screen in the same way as BLOADING does. BLOADING screens takes aprox. 4 seconds and CLOADING takes aprox. 30 seconds after the display has been found on tape. This may seem long as compared with disk but it is quicker than the other versions I discribed earlier.

I use a small machine code routine to capture the command CLOAD and it is then processed by my new CLOAD routine. This new routine then loads from tape ONLY BASIC PROGRAMS that start at 28672 decimal (7000H) ALL OTHER PROGRAMS ARE IGNORED. So after using this program the computer should be reset to use CLOAD in the normal manner.

LISTING 1 :-

```
10 POKE30884,151:POKE30885,123
20 POKE30969,153:POKE30970,123
```

Type in listing 1 and run. Listing 1 sets the start and end of program to a new start and end to allow for the machine code to be placed at the start of the program so as to allow the machine code to be saved along with the basic program. Normally the start of a basic program is 31465 decimal, it is now shifted to 31639.

Line 10 shifts the start address
Line 20 shifts the end address

LISTING 2 :-

```
10 FOR I = 31463 TO 31639:READ A :POKE I,A:B=B+A:NEXT
20 IF B <> 19235 THEN PRINT "ERROR IN DATA":END
30 NEW
40 DATA 0,0,9,123,0,0,177,51,48,56,54,50,44,49,51,58,177,51,48
50 DATA 56,54,51,44,49,50,51,58,88,213,193,40,88,41,0,0,0,0,,0
60 DATA 42,4,120,34,69,123,33,42,123,34,4,120,17,151,123,237
70 DATA 83,164,120,33,38,123,195,233,54,142,0,0,0,217,33,91,29
80 DATA 209,183,237,82,213,217,194,120,29,229,205,120,29,32,2
90 DATA 209,201,254,185,40,4,225,195,147,66,35,209,209,17,30,29
100 DATA 213,229,33,57,120,203,182,203,158,225,243,205,140,53,229
110 DATA 205,177,53,33,66,56,205,244,55,205,231,53,58,210,122
120 DATA 254,240,32,246,33,96,56,205,4,56,221,33,35,120,205
130 DATA 104,56,56,231,122,254,112,32,226,205,115,63,245,122
140 DATA 254,120,40,5,241,18,19,24,242,241,225,251,201,0,0,0
```

Type in listing 2 and RUN. LISTING 2 is for the capture and execution of the command CLOAD.

LISTING 3 :-

```
100 MODE(1)
110 POKE31273,PEEK(30884):POKE31274,PEEK(30885)
120 POKE31275,PEEK(30969):POKE31276,PEEK(30970)
130 A$="":A$=INKEY$:A$=INKEY$:IFA$=""THEN120
140 IFA$="S"THENSOUND20,1:GOSUB1000
```

```
150 IFA$="L"THENSOUND20,1:GOSUB1050
160 IFA$="C"THENMODE(1)
170 GOTO120
1000 POKE30884,0:POKE30885,112:POKE30969,0:POKE30970,120
1010 CSAVE"PICTURES"
1020 POKE30884,PEEK(31273):POKE30885,PEEK(31274)
1030 POKE30969,PEEK(31475):POKE30970,PEEK(31476)
1040 SOUND20,1:RETURN
1050 POKE30796,1:CLOAD
1060 POKE30796,0:SOUND20,1:RETURN
```

Type in listing 3 but do not run. Listing 3 is a small routine for CSAVING and CLOADING from tape. It is only the bare essentials. Screen dumps to printers, etc I leave to someone esle. Disk users could add disk saving and loading features to it as well. The following is how I transfered screen dumps from disk to tape.

```
1 GOTO 100
10 MODE(1):BLOAD"PICTURE"
```

Everytime I wanted to save a display from disk I typed the name of the display into line 10 and RUN 10. NO instructions appear on the screen as it goes to HI RES, (MODE 1).

S key SAVES the display to tape
L key LOADS a display from tape
C key clears the MODE(1) screen

LISTING 3 EXPLANATION :-

LINE 100 MODE(1) HI RES.
LINE 110 saves the start address of the program
LINE 120 saves the end address of the program
LINE 130 - 170 looks for an input from the keyboard
LINE 1000 sets up the start & end addresses for saving the screen
LINE 1010 CSAVES
LINE 1020 -1030 resets the start & end addresses
LINE 1040 makes a noise and returns for another input from the keyboard
LINE 1050 POKE 30796,1 disables the waiting & loading messages at the bottom of the screen. the only tape message printed if it happens is the loading error.
LINE 1060 resets 30796 & makes a noise & returns for another keyboard input.

The sound commands in the routine are important as we have no other way of knowing if the loading and especially the saving routines are finnished.

After entering listing 3 we are almost ready to csave the program but first we must change the start address back to 31465 WITH NO LINE NUMBERS TYPE :-

POKE 30884,233:POKE 30885,122 and then CSAVE or SAVE the program.

The CLOAD (machine code) routine is saved along with the basic program Entering more lines is no problem just type away. OR after you have a number of screens saved on tape you could use a FOR NEXT loop to load all the displays one after another.

EG. :- POKE30796,1:MODE(1):FORI=1TO N :CLOAD:NEXT
N is the number of displays saved on tape.

MOTOR SPORT

```
2 CLS:GOSUB3690:GOTO3222
3 GOSUB3600:TP=0:LK=1:QN=150:QM=150:CU=0:FL=9:GOTO2000
812 TS=TS+10:IFTS=3000GOSUB1050
813 IFTS=2930THENFS=1
820 IFPEEK(CF)=FLORPEEK(CF+1)=FLGOTO1100
822 IFPEEK(CF+32)=FLORPEEK(CF+33)=FLGOTO1100
823 IFPEEK(CF+32)=QN OR PEEK(CF+33)=QN THEN GOSUB 1056
825 POKECF,134:POKECF+1,137:POKECF+32,137:POKECF+33,134
826 SC(DV)=SC(DV)+.5
830 RL=29152+K
850 POKERL,FL:POKERL+L,FL
856 IFFS=1THENGOTO1000
858 IFTS>BRANDBP<10THEN1750
859 IFTS>DRANDBP<75THEN1360
860 IFTS>SSANDBP<70THEN1600
869 D=RND(10):P=INT(RND(20))
870 IFD<5K=K+1
875 IFD=5THENK=K+1
880 IFD>5K=K-1
890 IFK<2K=K+1
900 IFK>18K=K-1
920 IFP<10THENL=L-1
930 IFP>10THENL=L+1
935 IFL<7THENL=L+1
940 IFL>8THENL=L-1
1000 IFPEEK(26863)=247THEN1030
1020 IFPEEK(26863)=223THEN1035
1024 IFPEEK(26815)=239THENGOSUB3490
1028 GOTO1037
1030 CE=CF:CF=CF+1:GOTO1038
1035 CE=CF:CF=CF-1:GOTO1038
1037 CE=CF
1038 POKECE,96:POKECE+1,96:POKECE+32,96:POKECE+33,96
1039 PRINT@511," ";
1040 GOTO812
1050 FORPZ=RL-33+1TORL+L-32:POKEPZ,QN:NEXT
1055 RETURN
1056 FORM=0TO2
1057 SOUND17,1;22,1;26,1
1058 NEXT:TR=0:TU=1:GOSUB2305
1059 TS=0:POKECF,96:POKECF+1,96:POKECF+32,96
1060 POKECF+33,96
1070 FS=0:BP=0
1080 IFCF<29024THENCF=CF+64
1090 RETURN
1100 POKECF,142:POKECF+2,142:POKECF+32,142:POKECF+33,131
1102 PRINT@10,"  CRASH  ";
1103 POKECF+34,134
1104 GH=191
1110 FORYU=1TO5
1111 POKECF-32,GH:POKECF-1,GH:POKECF,GH:POKECF+1,GH:POKECF+32,GH
1113 SOUND30,2;20,2
1115 IFGH=191THENGH=159:NEXTYU:GOTO1110
1116 GH=191:NEXTYU
1118 FORUY=1TO200:NEXTUY:CLS
1127 PRINT@33,"A COMPLETE WRITE OFF THAT WHEEL"
1128 PRINT@(X+99),"G A M E  O V E R  "
1130 FORT=0TO1000:NEXT
1135 PRINT:PRINT
```

```
1136 IFSC(DV)=1THENPRINT"YOU WENT ";SC(DV);" MILES!":GOTO1150
1137 PRINT"YOU WENT ";SC(DV);" MILES!"
1150 FORTY=1TO700:NEXT:GOSUB3010:TS=0
1151 DV=DV+1
1152 IFDV>DETHEN3055
1160 TR=0:FORZ=0TO700:NEXT
1170 GOTO3
1210 CLS:COLOR2
1211 PRINT"   OK THEN,AND HAPPY MOTORING!"
1219 FORC=0TO100:COLOR1,1:COLOR,0:NEXT:SOUND31,3;12,2:STOP
1220 FORX=1TO8
1230 IFPEEK(28971+X)=134THENGOTO2515
1235 NEXT
1240 PRINT@384,"[illegible]"
1250 PRINT@416,"[illegible]"
1290 FORP=0TO1000:NEXT:COLOR6
1291 PRINT@384,"      ":PRINT@416,"      "
1292 POKECF,96:POKECF+1,96:POKECF+32,96
1293 POKECF+33,96
1300 GOTO2444
1360 IFL<6THENBP=BP+1:GOTO1000
1380 L=L-1
1390 GOTO1000
1600 IFL=7THEN1640
1610 IFL=8THEN1680
1620 L=L-1
1630 GOTO1000
1640 IFK>20THENL=8
1650 K=K+1
1660 BP=BP+1
1670 GOTO1000
1680 IFK<8THENL=7
1690 K=K-1
1700 BP=BP+1
1710 GOTO1000
1750 IFL=5THENBP=BP+1:GOTO1000
1760 L=L-1
1770 GOTO1000
2000 POKE30744,0:TP=0
2010 POKE30862,80:POKE30863,52
2300 CLS
2302 PRINT@22,"[illegible] ";DV
2303 PRINT@54,SC$(DV)
2305 DR=2000
2306 BR=600
2308 SS=1500
2310 IFTU=1THENTU=0:RETURN
2311 FORR=0TO15:FORS=10TO20STEP10
2312 RL=28672
2330 IFR=8ANDS=20THENGOTO2350
2335 IFR=9ANDS=20THENGOTO2350
2340 IFR=10ANDS=20THENGOSUB2360
2347 POKERL+32*R+S,FL
2350 NEXT:NEXT
2355 GOTO2410
2360 FORSZ=1TO10:POKERL+32*R+10+SZ+32,QN
2365 NEXT
2370 RETURN
2410 CF=28956
2411 AJ=0
2444 AJ=0:AG=0:COLOR2:CF=28956
```

CONTINUED NEXT ISSUE

STANDARD MICROSOFT / MACHINE	INKEY$ Returns character typed at keyboard or null if no character typed.	INPUT Read data from terminal.	INT Evaluates expression for largest integer contained.	LEFT$ Returns specified no. of characters starting at beginning of string.	LEN Gives decimal length of string.	LET Gives a value to a variable.	LIST List specified program lines at terminal.	LLIST List specified program lines at printer.	LOAD Load a program file into memory.	LOG Natural logarithm of expression.	MID$ Gives specified of characters to the right of start position in string.
	INKEYS	INPUT [STRING;] var [,var . . .]	INT(exp)	LEFTS(string, length)	LEN (string)	[LET] var=exp	list [lineno, lineno]	LLIST [lineno, lineno]	LOAD ["filename"]	LOG(exp)	MIDS(string, [,length])
APPLESOFT	GET VAR	INPUT[STRING,] VAR [,VAR . . .]	INT(exp)	LEFTS (string) LENGTH)	LEN(string)	[LET] var = exp	LIST [lineno, lineno] Note: '—' may be used in place of ','	[depends on interface arrangement—usually LIST"P"]	LOAD FILENAME	LOG(exp)	MIDS(string, start[,length])
ATARI		INPUT [exp] var [,var . . .] or INPUT [exp] string-var	INT(exp)	string (start, length)	LEN(string)	[LET] var=exp	LIST [lineno, lineno]	LIST "P"	CLOAD ["filename"][cass] or LOAD "filens:filename" [disk]	LOG(exp)	string(start [,length])
BBC MICRO	GET var [unlimited time] or INKEYS (time) Note: 100ths sec.	INPUT [string [,]] var [,var . . .]	INT(exp)	LEFTS(string, length)	LEN(string)	[LET] var=exp	LIST [lineno-lineno]	CTRL-B then LIST [lineno-lineno]	LOAD "filename" Note: "DISK" or "TAPE" to select device	LN(exp) NB: LOG(exp) gives common rather than natural log	MIDS(string, start[,length])
COMMODORE 64	GET var	INPUT [string;] var [,var . . .]	INT(exp)	LEFTS(string, length)	LEN(string)	[LET] var = exp	LIST [lineno-lineno]	OPEN 4,4: CMD4: LIST [lineno-lineno]	LOAD ["filename"] [cass] or LOAD "filename" 8 [disk]	LOG(exp)	MIDS(string, start[,length])
MICROBEE	KEY	INPUT (string) var (,var)	INT(real-exp)	var(:1, length)	LEN(string)	(LET) var=exp LET obligatory after THEN and ELSE	LIST (lineno. (,lineno)) forceloads	LLIST (lineno. (,lineno))	LOAD (U) (?) ("filename") LOAD U	LOG(real - exp)	var(:n,n+m-1) -n-start characters m-length
PET	GET var	INPUT [STRING, var [,var . . .]	INT(exp)	LEFTS (string, length)	LEN(string)	[LET] var = exp	LIST [Lineno-lineno]	OPEN 4.4: CMD4: LIST [lineno-lineno]	LOAD("filename"] [cass] or LOAD "filename", 8 [disk]	LOG(exp)	MIDS(string, start[,length])
TRS-80/SYSTEM 80	INKEYS	INPUT [string,] var [,var . . .]	INT(exp)	LEFTS(string, length)	LEN string)	[LET] var = exp	LIST [lineno-lineno]	LLIST [Lineno-lineno]	CLOAD "[filename]" [cass] or LOAD "filename" [disk floppy tape]	LOG(exp)	MIDS(string, start,length])
VIC-20	GET var	INPUT [string;] var [,var . . .]	INT(exp)	LEFTS(string, length)	LEN(string)	[LET] var = exp	LIST [lineno-lineno]	OPEN3,4:CMD 3: LIST [lineno-lineno]	LOAD ["filename"][cass] or LOAD "filename", 8 [disk]	LOG(exp)	MIDS(string, start[,length])
VZ200	INKEYS	INPUT[string,] var [,var . . .]	INT(exp)	LEFTS (string, length)	LEN (string)	[LET] var = exp	LIST [lineno-lineno]	LLIST [lineno-lineno]	CLOAD ["filename"]	LOG (exp)	MIDS (string, start, [,len])
ZX81	INKEYS	INPUT var	INT(exp)	string(TO finish)	LEN(string)	LET var = exp	LIST [lineno]	LLIST [lineno]	LOAD "[filename"]	LN(exp)	string(start TO finish)
ZX SPECTRUM	INKEYS	INPUT [string,] var	INT(exp)	string (TO finish)	LEN(string)	LET var = exp	LIST [lineno] Note: will fill screen then ask SCROLL?	LLIST [lineno]	LOAD "filename" [cass] Note: Microdrive manual for disk	LN(exp)	string(start TO finish)